Title	:	HIGH-LEVEL PROGRAM INTERFACE FOR GRAPHICS OPERATIONS			
Inventors	:	JOHN HARPER et al			
Serial No.	:	10/826,762	Filed	:	April 16, 2004
Examiner	:	Joni Hsu	Art Unit	:	2628
Docket	:	119-0033US	Customer	:	61947

Mail Stop Appeal Brief Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

I. REAL PARTY IN INTEREST

Apple Inc. is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

None.

III. STATUS OF CLAIMS

Claims 1-85 are rejected. Claims 1-85 are appealed.

IV. STATUS OF AMENDMENTS

None.

V. SUMMARY OF CLAIMED SUBJECT MATTER

This section provides a concise explanation of the subject matter defined in each independent claim involved in this appeal, referring to the specification by paragraph and line number and to the drawings by reference characters as required by 37 C.F.R. 41.37(c)(I)(v). It should be noted, citation to passages in the specification and drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element. All references to page and line numbers below are per the specification as filed on 21 May 2002 and amendments to claims filed on 14 Feb 2006. Additionally, references are not necessarily exhaustive, and various claim elements may also be described at other locations.

Generally, Appellant claims methods (independent claims 1, 28, 43, 58, 63, 74, and 75), a system (independent claim 22), an application program interface (independent claims 79 and 85) and a computer-readable medium (independent claim 86) to provide a high-level program interface for graphics operations.

Independent claim 1 recites a method of editing an initial image (¶ 43, 8. 1-3), the method comprising the steps of:

- A first process requesting a filter (¶ 44, 9. 1-7) from a second process;
- said first process defining a relationship (¶ 51, 10. 8-10) between said filter (¶ 44, 9. 1-7) and said initial image (¶ 43, 8. 1-3), said related filter (¶ 44, 9. 1-7) and initial image (¶ 43, 8. 1-3) comprising a program;
- said second process compiling said program, yielding a compiled program;
- running at least a portion of said compiled program to apply a function of said filter (¶ 44, 9. 1-7) to said image (¶ 43, 8. 1-3), yielding an pixel-image (¶ 70, 17. 15-16) result.

Independent claim 22 recites a system for editing an initial image (¶ 43, 8. 1-3), comprising:

- A first microprocessor (Fig. 1, element 11) running a first process and a second process for servicing requests from said first process;
- a memory (Fig. 1, element 114) for storing a filter (¶ 44, 9. 1-7), said filter (¶ 44, 9. 1-7) requested by said first process;
- a second memory (Fig. 1, element 114) for storing a data structure (¶ 51, 10. 5-7) comprising a relationship (¶ 51, 10. 8-10) between said initial image (¶ 43, 8. 1-3) and said filter (¶ 44, 9. 1-7), said first process causing the creation of said data structure (¶ 51, 10. 5-7);
- a second microprocessor (Fig. 1, element 112) for running a program created using said data structure (¶ 51, 10. 5-7);
- a third memory (Fig. 1, element 114 or element 113) for storing a pixel image resulting (¶ 70, 17. 15-16) from running said program.

Independent claim 28 recites a method of creating a result image (¶ 70, 17. 15-16) comprising the steps of:

- a first process requesting the creation of a context (¶ 45, 9. 1-3);
- said first process requesting the creation of a result image (¶ 70, 17. 15-16);
- said first process indicating parameters associated with the creation of said result image (¶ 70, 17. 15-16);
- said first process requesting that said result image (¶ 70, 17. 15-16) be rendered to said context (¶ 45, 9. 1-3);
- a second process servicing said requests of said first process, said servicing comprising the steps of:

- optimizing a graph (¶ 70, 17. 3-13; Fig. 4) representing said result image (¶ 70, 17. 15-16);
- compiling said optimized graph (¶ 70, 17. 12-15);
- causing the rendering of said compiled optimized graph into said context (¶ 70, 17. 15-16).

Independent claim 43 recites a method of creating an result image (¶ 70, 17. 15-16) comprising the steps of:

- A first process running on a first microprocessor (Fig. 1, element 11) requesting the creation of a context (¶ 55, 11. 1-4);
- said first process (¶ 34, 6. 1-12) requesting the creation of a result image (¶ 70, 17. 15-16);
- said first process indicating parameters (¶ 53, 11. 1-2; Fig. 5, examples) associated with the creation of said result image (¶ 70, 17. 15-16);
- said first process requesting that said result image (¶ 70, 17. 15-16) be rendered to said context (¶ 45, 9. 1-3);
- a second process running on said first microprocessor (Fig. 1, element 11) servicing said requests of said first process, said servicing comprising the steps of:
  - optimizing a graph (¶ 36, 7. 1-11; Fig. 4) representing said result image (¶ 70, 17. 15-16);
  - compiling said optimized graph (¶ 53, 11. 1-12);
  - causing rendering of said compiled optimized graph into said context (¶ 53, 11. 1-12), said rendering occurring on a second microprocessor (Fig. 1, element 112).

Independent claim 58 recites a method for providing a high level interface to a graphics processing resource comprising:

- A first process requesting performance of a task through one or more function calls serviced by said graphics processing resource (¶ 33-34, 6.; Fig. 3, element C101), said function calls selected from one or more of the following options, (i) creating of an image (¶ 57, 12. 1-7), (ii) creating a filter (¶¶ 59-60, 13. all), (iii) setting arguments associated with said filter (¶ 65, 16. 1-5), (iv) asking for the output of said filter (¶ 67, 16. 1-6) or another filter, (v) creating a context (¶ 55, 11. 1-5); and (vi) rendering an image (¶ 70, 17. 1-3) to said context or another context (¶ 45, 9. 1-3);

- said request having an object associated therewith, said object comprising one of the following: a context (¶ 45, 9. 1-3), an image (¶ 43, 8. 1-3), a filter (¶ 44, 9. 1-7), or a vector (¶ 46, 9. 1-6);

- said request defining a relationship (¶ 51, 10. 8-10) between at least one of said functions and one of said objects;

- said graphics processing resource (¶ 33-34, 6.; Fig. 3, element C101) servicing said request.

Independent claim 63 recites a method for providing a high level interface to a graphics processing resource comprising:

- A first process requesting performance of a task through one or more function calls serviced by said graphics processing resource (¶ 33-34, 6.; Fig. 3, element C101), said function calls selected from one or more of the following options, (i) creating of an image (¶ 57, 12. 1-7), (ii) creating a context (¶ 55-56, 11-12. all); and (v) rendering an image (¶ 43, 8. 1-3) to said context (¶ 70, 17. 1-16) or another context (¶ 45, 9. 1-3);

- said request having an object associated therewith, said object comprising one of the following: a context (¶ 45, 9. 1-3), or an image (¶ 43, 8. 1-3);

- said request defining a relationship (¶ 51, 10. 8-10) between at least one of said functions and one of said object;

- said graphics processing resource (¶ 33-34, 6.; Fig. 3, element C101) servicing said request.

Independent claim 74 recites a method for rendering an image (¶ 70-71, 17. all) comprising the steps of:

- A first process running on a CPU (¶ 9, 3. 1-12; Fig. 1, element 11) requesting the creation of an image (¶ 57-58, 12. all);

- a graphics services resource (¶ 33-34, 6.; Fig. 3, element C101), responding to said request by running a first routine on said CPU (¶ 9, 3. 1-12; Fig. 1, element 11), said first routine for optimizing a graph (¶ 70, 17. 3-15; Fig. 4) representing said image (¶ 43, 8. 1-3);

- said first process requesting the rendering of said image (¶ 70-71, 17. all) to a specified context (¶ 45, 9. 1-3);

- said graphics services resource (¶ 33-34, 6.; Fig. 3, element C101) causing the rendering of said graph (¶ 70, 17. 15-16; Fig. 4) representing said image (¶ 43, 8. 1-3), said rendering occurring on a GPU (¶ 9, 3. 1-12; Fig. 1, element 112).

Independent claim 75 recites a method for converting a first image (¶ 43, 8. 1-3) representation into a second image (¶ 43, 8. 1-3) representation, comprising the steps of:

- Creating a first graph (¶ 36, 7. 1-11; Fig. 4) associated with said first image (¶ 43, 8. 1-3) representation where software routines for creating such graph (¶ 36, 7. 1-11; Fig. 4) execute on a CPU (¶ 9, 3. 1-12; Fig. 1, element 11);

- determining an intersection of the first graph's global domain of definition (¶ 92, 24. 1-8) and global region of interest (¶ 98, 26. 1-11);

- resolving a first node in said first graph by running software routines on said CPU (¶ 9, 3. 1-12; Fig. 1, element 11) to (i) determine if said first node may be combined with a second node and (ii) create program steps for calculating and storing only the portions of any intermediary image (¶ 98, 26. 8-14) that relate to said intersection (¶ 101, 27. 8-10), said program steps for compilation on said CPU (¶ 9, 3. 1-12; Fig. 1, element 11) and execution on a GPU (¶ 9, 3. 1-12; Fig. 1, element 112).

Independent claim 79 recites an image (¶ 43, 8. 1-3) processing application program interface embodied on one or more computer readable media, comprising: a first group of services related to filter objects (¶ 59, 13. 1-9); a second group of services related to image objects (¶ 57-58, 12. all); a third group of services related to context objects (¶ 55, 11. 1-5); and a fourth group of services related to vector objects (¶ 65, 16. 3-5).

Independent claim 85 recites an application program interface for facilitating image (¶ 43, 8. 1-3) processing, the application program interface comprising functions to: create image objects (¶ 57-58, 12. all); create context objects (¶ 55-56, 11-12. all); create filter objects (¶ 59, 13. 1-9; Fig. 5 , element 52); set filter object parameters (¶ 65, 16. 1-9); obtain filter object output (¶67, 16. 1-4); and convert image objects into context objects (¶ 70, 17. 1-16).

Independent claim 86 recites a computer-readable medium having computer executable instructions for performing the method recited in any one of claims 1, 28, 43, 58, 63, 74 or 75.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claim 75 stands rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent 6,717,599 B1 to Olano ("Olano").

Claims 79-84 stand rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent 6,600,840 B1 to McCrossin et al. ("McCrossin).

Claims 1, 2, 4-6, 8, 11-23, 25-32, 36-41, 43-47, 51-56, 58, 61-64, 66-74, and 85 stand rejected under 35 U.S.C. 103(a) as obvious over Olano in view of McCrossin.

Claims 3, 33-35, and 48-50 are rejected under 35 U.S.C. 103(a) as obvious over Olano and McCrossin in view of U.S. Patent Application Publication US 2002/0033844 A1 to Levy et al. ("Levy").

Claims 7, 9, 60, and 65 stand rejected under 35 U.S.C. 103(a) as being obvious over Olano and McCrossin in view of U.S. Patent 6,411,301 to Parikh et al. ("Parikh").

Claim 10 stands rejected under 35 U.S.C. 103(a) as being obvious over Olano, McCrossin, and Parikh in view of U.S. Patent 6,867,799 B1 to Doyle et al. ("Doyle").

Claim 24 stands rejected under 35 U.S.C. 103(a) as being obvious over Olano and McCrossin in view of U.S. Patent 5,854,637 to Sturges ("Sturges").

Claims 42 and 57 are rejected under 35 U.S.C. 103(a) as being obvious over Olano and McCrossin in view of U.S. Patent 6,977,661 B1 to Stokes et al. ("Stokes").

Claim 59 stands rejected under 35 U.S.C. 103(a) as being obvious over Olano and McCrossin in view of Doyle.

Claims 76 and 77 stand rejected under 35 U.S.C. 103(a) as being obvious over Olano in view of Levy.

Claim 78 stands rejected under 35 U.S.C. 103(a) as being obvious over Olano in view of U.S. Patent 6,266,053 to French et al. ("French").

VII. ARGUMENT

All the claims do not stand or fall together. Instead, appellants present separate arguments for various groups of independent and dependent claims. Any claim argued separately is under a subheading identifying the claim by number. Claims argued as a group are under a subheading identifying the claims by number. After a concise discussion of the cited art, each of these arguments is separately presented below under separate headings and sub-heading as required by 37 C.F.R. 41.37(c)(l)(vii).

Legal Principles

Even though the Examiner is entitled to give the claims their broadest reasonable interpretation (See M.P.E.P. 2111), the Examiner cannot read the claim language so broadly as to completely ignore the explicit language actually recited in the claims. In fact, each word in Appellant's claim must be considered in determining whether the claims are patentable over the cited prior art. See M.P.E.P. 2143.03. In addition, every element of Appellant's claims must be identically shown in the cited prior art for the cited prior art to anticipate in terms of 35 U.S.C. 102. See Diversitech Corp. v. Century Steps, Inc., 850 F.2d 675, 677, 7 U.S.P.Q.2d 1315, 1317 (Fed. Cir. 1988); See also Richardson v. Suzuki Motor Co., 868 F.2d 1226, 1236, 9 U.S.P.Q.2d 1913, 1920 (Fed. Cir. 1989), cert. denied, 493 U.S. 853 (1989) (The "identical invention must be shown in as complete detail as is contained in the patent claim"). Therefore, as will be explained below the unreasoned combination of Olano and McCrossin does not disclose each element contained in the claims, and there is substantial difference between the claimed invention and the disclosures of Olano and McCrossin. For each of the Examiner's rejections below Appellant will show that the Examiner has largely taken the references out of context or severely extrapolated from those references and then proceeded to combine them in an unreasoned manner.

A. Section 102 Rejections (35 U.S.C. 102)

US Patent 6,717,599 to Olano

The Examiner primarily relies on US. Patent 6,717,599 issued to Olano. Appellant disagrees with the Examiner's application of Olano because Olano bears no substantive analogy to the claimed inventions.

Olano proposes a system for implementing mathematical derivatives using graphics hardware. The entirety of the Olano disclosure centers around a single method for implementing mathematical derivatives and that method is shown in figure 3. The method specifies: receiving "a data structure that represents the operation of one or more program statements, ***wherein at least one of the program statements requires derivative information***"; marking "a node of the

data structure ***that determines derivative information***"; and applying " a transform rule to the node marked in step 320 to obtain a transformed data structure, wherein the operations represented by the transformed data structure can be implemented with graphics hardware interface program." See Figure 4, emphasis added. In accordance with this single method, Olano's compiler transforms program statements into a tree structure comprising nodes that are either single operators or single variables. Olano's system then performs a rule look-up to determine how a specific tree-segment can be transformed so that the involved derivative math can be performed by resident graphics hardware. Thus, in essence, Olano's system merely suggests how to use a low-level tree and rule look-up to perform derivative math on a graphics processor. Olano's essence is not concerned with images or editing images.

US Patent 6,600,840 to McCrossin

The Examiner also relies heavily on U.S. Patent 6,600,840 issued to McCrossin. Appellant also disagrees with the Examiner's application of McCrossin, because like Olano, McCrossin bears no analogy to the claimed inventions.

McCrossin proposes a system for changing the format of an image by applying a series of filters. The filters are applied to the source image for each parameter of the source image that does not match a parameter of the requested or desired result image. In essence, McCrossin proposes a system to simplify the conversion of the ***same*** image from one ***format*** to another; for example from a BMP file to a TIFF file. See for example Col. 6 line 40 et. Seq. Thus, McCrossin relates to changing the format of an image, not editing the image! The whole point of McCrossin is to maintain the **same** image while changing only the format. That McCrossin manipulates image bits to make the conversion is essentially a technicality that does not make McCrossin a reference pertaining to image creation and editing.

1. Claim 75

The Examiner has specifically rejected claim 75 as anticipated by Olano based on several inappropriate inferences and extrapolations. Firstly, there is nothing in Olano's disclosure that fairly teaches or suggests the expressly claimed division of work for a CPU and a GPU. The

disclosure in Olano is for a general purpose computer having a graphics processor and a CPU, each performing their standard functions. Olano discloses that special purpose graphics processors and graphics hardware are designed to process millions of pixels each second. Thus Olano reasons that derivative operations can be implemented at an interactive rate on available computer graphics systems. (see Olano at Col. 2 lines 25-45). Olano further discloses how to implement "any derivative operator in graphics hardware" at Col 7 lines 49-50. Thus the disclosure in Olano is directed to extending the number and type of mathematical operations that can be performed on the graphics hardware and not to the specific division of work as claimed in independent claim 75. Specifically, "Resolving a first node in said first graph by running software routines on said CPU to (i) determine if said first node may be combined with a second node and (ii) create program steps for calculating and storing only the portions of any intermediary image that relate to said intersection, said program steps for compilation on said CPU and execution on a GPU."

The Examiner further rejects claim 75 using disjointed references within Olano to contend that Olano teaches "a global domain of definition; a global region of interest; [and] the intersection of the foregoing two items" *See* Office Action dated 30 March 2007 at ¶ 3 on page 3-4. Contrary to the Examiners contention, "Domain of definition" and "region of interest" are image-related concepts that are totally absent in Olano. Olano is performing derivative math and "marking (or labeling) of the nodes of input 204 that require derivative information" in a math based data structure. *See* Olano Col. 5 lines 36-38. The image related concepts of domain of definition ("DOD") and region of interest ("ROI") are explained, for example, in ¶¶ 92-98 of appellants specification; but simply put, a DOD is "a representation of all places in an **image** that are explicitly defined and not transparent", while the ROI is "the portion of the **input image** that is necessary to compute the given output DOD." These concepts are explicitly directed to area within an **image** while, by dramatic contrast, the cited portion of Olano refers explicitly to mathematical reduction of a tree structure. This is clearly and definitively stated in Olano's invention summary and throughout Olano's disclosure:

> "The present invention provides a method, system, and computer program product for implementing derivative operators at an interactive rate in computer graphics

> systems. In an embodiment, ***a data structure representing the operation of one or more computer program statements*** is received by a compiler and transformed into a tree data structure. Nodes of the tree data structure that determine derivative information are ***marked***. A transform rule is applied to the marked nodes to transform the tree data structure into a data structure that can be implemented with graphics hardware interface program statements." Olano, Col. 2, lines 41 – 52 (emphasis added).

Therefore, it is clear that Olano proposes nothing even analogous to the claimed **image** related "DOD" and "ROI" limitations of claim 75.

The Examiner has further relied on Olano to teach "creating program steps relating to the aforementioned intersection [of ROI and DOD]." In particular, the Examiner cites the following: "input 204 contains shading language statements, input 204 is converted into a tree data structure representation by compiler 104, the nodes of the tree data structure representation are then converted into graphics API statements, reducing the data structure of input 204." (Olano Col. 5, lines 5-13, 35-36 and Office Action dated 30 March 2007 at ¶ 3 on page 3-4). This citation discusses reduction of the node tree for simplifying computation. It bears no analogy to the image related concept of intersecting DOD and ROI. The Examiner's citation has nothing to do with any kind of intersection and as explained above, Olano's tree structure contains "Nodes … that determine derivative information" and not the image information stated in the claim.

The Examiner has further rejected claim 75 as anticipated by Olano based on the Examiner's own inference that "the compiler must inherently not be a part of the graphics hardware … the compiler must inherently be processed by a CPU" *See* Office Action dated 30 March 2007 at ¶ 17 on page 15. This inference is inappropriate in view of Olano's description:

> "Compiler 104 is **similar to a compiler for a complex instruction set computer** because graphics API statements are similar to complex instructions. **One difference is that compiler 104 does not generally consider scheduling issues because there is no overlap between rending passes through a graphics pipeline**." Olano at Col 5, lines 15-18 emphasis added.

In the above passage, Olano is discussing a special implementation of a compiler and one skilled in the art would understand this compiler to be unique from the norm, thus obviating the Examiner's inference of inherency based upon the norm. In short, Olano discloses a "compiler 104 comprises a set of derivative transform rules 202 that are applied to an input 204 to produce an output 206" (Olano Col. 5 lines 3-5). This compiler is more of a translator than a standard compiler and the disclosure has no clear indication regarding the execution of this particular compiler. Furthermore, the Examiner states that "Since the graphics hardware can only operate on statements that are supported by standard graphics hardware, and the compiler is able to transform statements that are not supported by standard graphics hardware, the compiler must inherently not be a part of the graphics hardware." *See* Office Action dated 30 March 2007 at ¶ 17 on page 15. The Examiner seems to contend it is impossible for graphics hardware to run a compiler. The Examiner offers no evidence of this contention. In addition, the contention seems particularly off-base for Olano, which teaches that graphics hardware may be used for a non-traditional purpose – derivative math. Even if the Examiner's inference is valid, claim 75 calls for much more than compilation by a compiler.

For all the reasons set forth above, claim 75 is clearly allowable over the cited art and thus currently in condition for allowance.

2. Claims 79 – 83

The Examiner has specifically rejected claim 79 as anticipated by McCrossin. As discussed earlier, McCrossin proposes a system for changing the format of an image by applying a series of filters. In essence, McCrossin proposes a system to simplify the conversion of the ***same*** image from one ***format*** to another; for example from a BMP file to a TIFF file. See for example Col. 6 line 40 et. Seq. Thus, the API discussed by McCrossin does not disclose an image processing API as set forth in claim 79 rather, McCrossin merely discloses common "read" and "write" APIs associated with, "an improved method and system for transforming image data from one format to another wherein the number of converters required is reduced." McCrossin at Col. 1 lines 19-21. The Examiner cites to, "An application (not shown) calls the transform object using a procedure call or API call, which results in call 166 being made." (McCrossin at Col. 7, lines 33-35). As can be clearly seen in Figures 8A and 8B, element 166 is

a common read function call (fread). Therefore, McCrossin is not teaching any kind of API here other than a common "read" API that performs a read function on an image whose format will be changed under McCrossin's principles. The Examiner is incorrect in inferring that McCrossin teaches the specific bundle of API services as claimed by merely stating that an API call or procedure call is used. Simply put, the disclosure in McCrossin comprises only read and write APIs and a translator of image formats. In dramatic contrast, Appellant's invention in claim 79 is for a specifically claimed "image processing application program interface;" i.e. APIs to filter objects, APIs to image objects, APIs to context objects and APIs to vector objects. These two concepts are abundantly different.

The Examiner also states that "Transform object 103 determines an actual image vector 139, which describes the format of the image to be read or written" (McCrossin at Col. 6, lines 51-53). The Examiner then states "Since this describes the format of the image, this is related to image objects and context objects." (See Office Action dated 30 March 2007 at ¶ 19 page 18). Again the Examiner has made an incorrect extrapolation, simply describing the format of the image does **not** encompass the requirements of image objects and context objects. As will be seen in the discussion of claim 84 below, the Examiner appears to have an incorrect understanding of the "context objects" as disclosed and claimed by Appellant. For example, in the Office Action dated 30 March 2007 at ¶ 24 page 19, the Examiner misquotes Appellant's specification to say, "creating a context is derived from tools that allow the definition of an object in memory." The correct quotation from Appellant's specification is "the **ability** to create a context is derived from tools that allow definition of an object in memory." This simply means that Appellant is teaching that a context may be created using "tools that allow definition of an object in memory." The Examiner seems to have incorrectly interpreted the partial quotation to believe that **any** definition of an object in memory is creating a context. This is simply not true and not what is stated in Appellant's claims or disclosure. McCrossin does not teach or discuss context creation and the Examiner's belief to the contrary appears to be based on his belief that **any** definition of an object in memory is creating a context.

McCrossin does not and can not disclose a bundle of expressly claimed API services; i.e. API services related to filter objects, API services related to image objects, API services related

to context objects and API services related to vector objects. Such a bundle of API services as claimed is entirely absent from McCrossin at least because a major purpose of McCrossin is to **isolate** the application layer from the task of doing a file format conversion. For example, the McCrossin application uses a common "read" or "write" API to obtain a file that is in a format associated with the isolated application program, McCrossin's API does NOT call any or all "filters" involved in the file format conversion). The Examiner has found a single reference to "a procedure call or API call" in McCrossin at Col. 7, lines 33-35 and inappropriately extrapolated this reference to contend that McCrossin teaches the specifically claimed limitations of independent claim 79. Thus, in view of this clarification, appellant submits that claim 79 should be in condition for allowance. Claims 80 through 83 depend from claim 79 and therefore should be allowable for the same reasons.

3. Claim 84

Similar to claim 79, the Examiner asserts that claim 84 is invalid over McCrossin. Appellant submits that claim 84 is in condition for allowance because, as explained with respect to claim 79, McCrossin discloses only read and write APIs in the context of image format translation and does not and can not disclose, the specifically claimed a bundle of API functions; i.e. APIs to create image objects; APIs to create context objects; APIs to create filter objects; APIs to set filter object parameters; APIs to obtain filter object output; and APIs to convert image objects to context objects. In particular the Examiner cites McCrossin at Col. 6 lines 59-62, which clearly states that "Filters are accessed … from filter library". The Examiner has found a reference in McCrossin that discloses the application of a filter to an image. However, it is not valid for the Examiner to extrapolate that reference to an API to **create** filter objects rather than retrieve them from a pre-existing library. In the Office Action dated 30 March 2007 at ¶ 24 page 19, the Examiner misquotes Appellant's specification to say, "creating a context is derived from tools that allow the definition of an object in memory." The correct quotation from Appellant's specification is "the **ability** to create a context is derived from tools that allow definition of an object in memory." This simply means that Appellant is teaching that a context may be created using "tools that allow definition of an object in memory." The Examiner seems to have incorrectly interpreted the partial quotation to believe that **any** definition of an object in

memory is creating a context. This is simply not true and not what is stated in Appellant's disclosure. McCrossin does not teach or discuss context creation and the Examiner's belief to the contrary appears to be based on his belief that **any** definition of an object in memory is creating a context. For example, the Examiner states, "Since McCrossin discloses that the actual image vector describes the format of the image to be read or written, this allows definition of an object in memory, and therefore is considered to create context objects, and therefore McCrossin does teach API functions to create context objects." (Office Action dated 30 March 2007 at ¶ 24 pages 19-20). Finally, even if any object were a context object (and this is not true), McCrossin does not teach creation of the object through an API, as claimed. Recall, McCrossin's only APIs are to read and write. Thus, in view of this clarification, appellant submits that claim 84 should be in condition for allowance.

B. Section 103 Rejections (35 U.S.C. 103)

The board and the Examiner are reminded:

> To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some reason to combine or try to combine the reference teachings, considering the interrelated teachings of multiple patents, needs or problems known in the field of endeavor at the time of the invention, and the knowledge generally available to one of ordinary skill in the art. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. MPEP § 2143, as modified by KSR v. Teleflex, 127 S.Ct. 1727 (2007).

The Examiner has relied on varying combinations of eight different references as basis for rejection under 35 U.S.C. 103. To aid in understanding of the Applicant's arguments the eight references are briefly summarized here with attention to the Examiners various combinations. As will clearly be seen the Examiner has combined references that have no logical connection to each other. Since the references are not logically combined, there is no possibility of their teaching or suggesting anything reasonable. These eight references are disjointed and the only comprehensive relationship they have to each other is that they appear to have been found by a compound word search algorithm. Furthermore, to the extent that any single reference discloses a portion of Appellant's claim limitations, that portion is in isolation

and the references (alone or together) lack the claimed connectivity to the other elements. Throughout the section 103 rejections the Examiner ignores the absence of the claimed connectivity (and sometimes affirmative elements) in the cited combinations. This misapplication of the §103 rejections occurs throughout, however most acutely in the many three-way and four-way art combinations..

1. Olano US Patent 6,717,599

The Examiner's primary cited reference for virtually all the section 103 rejections is Olano. As discussed above, Olano proposes a system for implementing mathematical derivatives using graphics hardware. In particular, Olano's disclosure proposes a low-level tree structure and rule look-up used to perform derivative math on a GPU. For purposes of assessing the section 103 rejections, Appellant urges consideration of the true obscurity of this reference due to its solitary topic of derivative math. Olano is only tenuously related to normal graphics processing, and it lacks any reasonable analogy to the image processing systems and methods claimed by Appellant.

McCrossin (US Patent 6,600,840)

The Examiner also cites McCrossin in several section 103 art combinations. As discussed above, McCrossin proposes a system for changing the format of an image by applying a series of filters. The filters are applied to the source image for each parameter of the source image that does not match a parameter of the requested or desired result image. In essence, McCrossin proposes a system to simplify the conversion of the ***same*** image from one ***format*** to another; for example from a BMP file to a TIFF file. See for example Col. 6 line 40 et. Seq.

Olano and McCrossin

The Examiner has rejected claims 1, 2, 4-6, 11-23, 25-32, 36-41, 43-47, 51-56, 58, 61-64, 66-74 and 85 under Section 103 in view of the combination of Olano and McCrossin. As noted above this is combination of (i) a system for implementing mathematical derivatives on graphics hardware, with (ii) a system to convert the same image from one format to another. Neither McCrossin nor Olano is pertinent to the claimed inventions, which relate to methods and systems

for (i) editing an image using 2 cooperating processes; (ii) providing a graphics AP or resource for editing an image; or (iii) rendering an image. More importantly, there would be no reason to combine these references in that they, in no way, address the same problems as each other or Applicant's disclosure. The only relationship Applicant perceives in this cited art is that they were likely coincidentally responsive to the same key word search.

Olano, McCrossin and Levy (US Publication 20020033844A1)

The Examiner rejects claims 3, 33-35, and 48-50 as being obvious over the 3-way combination of Olano, McCrossin and Levy (US 20020033844A1). The Examiner relies on Levy to teach "the step of using a cache look-up". The disclosure in Levy (U.S. published application 20020033844) is directed to a method of connecting multimedia content to a network resource. Levy's proposal extracts an identifier from a media signal, such as a digital watermark, perceptual hash, or other machine extracted signal identifier and sends the identifier to a network along with context information indicating device type information. The related data is adapted to the network-connected device based on the device type information and formatted for rendering on the display type of the device. Ultimately, Levy relates to methods for associating multimedia signals with related data using identifiers associated with the signals. (Abstract, [0006]). Thus, The Examiner's citation to the three-way combination is as follows: (i) a specialized method regarding digital watermarks (Levy); (ii) a system for implementing mathematical derivatives on graphics hardware (Olano), and (iii) a system to convert the same image from one format to another (McCrossin). None of the three references is particularly pertinent to the claimed inventions. More importantly, there would be no reason to combine these references in that they, in no way, address the same problems as each other or Applicant's disclosure. Furthermore, the strained analogy of McCrossin and Olano is demonstrated in this three-way rejection because the Examiner needed a third reference to find a function as common as a cache look-up. This fact alone should demonstrate that Appellant's claims are clearly non-obvious over any combination of Olano and McCrossin. Nevertheless, there is no reasonable relationship between Olano, McCrossin and Levy that would suggest combination.

Olano, McCrossin and Parikh (US Patent 6,411,301)

The Examiner rejects claims 7, 9, 60 and 65 as obvious in view of the 3-way combination of Olano, McCrossin and Parikh (US 6,411,301). The Examiners relies on Parikh to teach "a compiled program for yielding a pixel-image result." The disclosure in Parikh (U.S. Pat. 6411301) is directed to an interface for a graphics system, which purportedly includes simple yet powerful constructs that are easy for an application programmer to use and learn. Features include unique vertex representation allowing the graphics pipeline to retain vertex state information and to mix indexed and direct vertex values and attributes; a projection matrix value set command; a display list call object command; and an embedded frame buffer clear/set command. This flexible representation allows the game developer to organize vertex data in a way that is appropriate for the game. The ability to index each component separately eliminates a great deal of data duplication. (Abstract, summary). As with Levy above, the Examiner's need of a third reference to show "a compiled program for yielding a pixel image result" emphasizes the inapplicability of Olano and McCrossin. Further, there would be no reason to combine Parikh with Olano and McCrossin in that these three references do not address problems either similar to each other or Applicant's disclosure. There is no reasonable relationship between these three references that would suggest combination.

Olano, McCrossin, Parikh and Doyle (US Patent 6,867,779)

The Examiner rejects claim 10 based upon the four-way combination of Olano, McCrossin, Parikh and Doyle. The Examiner also rejects claim 59 based upon the three-way combination of Olano, McCrossin and Doyle. The disclosure in Doyle (U.S. Pat. 6,867,779) is directed toward image rendering by dividing the image into chunks, rendering the chunks in one of at least two devices, and determining which of the devices renders each one of at least some of the chunks based on at least one device's progress in the rendering of other chunks. (Abstract). There would be no reason to combine Doyle with either (i) Parikh, Olano and McCrossin or, (ii) Olano and McCrossin. These references do not address the same problems as each other or as Applicant's disclosure. Indeed, if Appellant or anyone had combined three or four disparate references as these, the combination would certainly be non-obvious. There is no reasonable relationship among either of the three-way or four-way sets of art cited by the Examiner.

Olano, McCrossin and Sturges (US Patent 5,854,637)

The Examiner rejects claim 24 as obvious over the three-way combination of Olano, McCrossin and Sturges (US 5,854,637). Sturges is cited for its alleged teaching regarding a graphics API. The disclosure in Sturges is directed toward a computer system having a shared memory accessible by a memory controller and a graphics controller. A portion of the shared memory is selectively defined as a frame buffer. Graphics operations provided by a client device are routed through the memory controller if a frame buffer is created. If no frame buffer is created, the graphics commands are routed through the graphics controller. (Abstract) Sturges is trying to provide an improved method and apparatus for expediting the execution of graphics operations generated by the microprocessor within a system employing a shared memory. Sturges discloses memory management and interface techniques at the **hardware** level. The only reference of an API in Sturges is found at Col. 11 line 45 when the actual letters **API** show up in the context of a **company name**. This exemplifies the fact that the Examiner has used word search results to combine references that have no reasonable relationship between them. Like the other three-way combination regarding Olano and McCrossin, the Examiner's need for this third reference demonstrates the inapplicability of Olano and McCrossin. There would be no reason to combine Sturges, Olano and McCrossin in that they, in no way, address the same problems as each other or Applicant's disclosure. There is no reasonable relationship between these three references that would suggest combination.

Olano, McCrossin and Stokes (US Patent 6,977661)

The Examiner rejects claims 42 and 57 as obvious over the three-way combination of Olano, McCrossin and Stokes (U.S. Pat. 6,977,661). The disclosure in Stokes is directed toward a system and method for image acquisition, which enables selective automated application of color management to color image data generated by an image-capturing device, by the device driver for the device. Like the other three-way combinations involving Olano and McCrossin, there would be no reason to combine with Stokes because the three references do not address the same problems as each other or as Applicant's disclosure. There is no reasonable relationship between these three references that would suggest combination.

Olano and French (US Patent 6,266,053)

Finally, the Examiner rejects claim 78 based upon the combination of Olano and French (U.S. Pat. 6,266,053). The disclosure in French is directed toward a technique for representing a visual scene as a directed acyclic graph of data and operators that generates a sequence of image frames over specified time intervals. The graph specifies temporal and spatial values for associated visual elements of the scene. Time is modeled in the inheritance properties explicitly defined within the scene graph hierarchy, by assigning temporal attributes to each media element. There would be no reason to combine French with Olano in that the two references do not address the same problems as each other or as Applicant's disclosure. There is no reasonable relationship between these three references that would suggest combination.

As can be clearly seen, each of the eight cited references vary widely from Appellant's invention. Appellant submits that there is no reason to combine references that differ as much as the cited combination and even if these references are combined they do not teach or suggest the comprehensive set elements claimed by Appellant.

C. Rejections In View Of The Combination Of Olano And Mccrossin

The Examiner has rejected **claims 1, 2, 4-6, 11-23, 25-32, 36-41, 43-47, 51-56, 58, 61-64, 66-74 and 85** under Section 103 in view of the combination of Olano and McCrossin. As noted above, this is a combination of (i) a system for implementing mathematical derivatives on graphics hardware, with (ii) a system to convert the same image from one format to another. Neither McCrossin nor Olano is pertinent to the claimed inventions, which relate to methods and systems for (i) editing an image using two cooperating processes; (ii) providing a graphics AP or resource for editing an image; or (iii) rendering an image. More importantly, there would be no reason to combine Olano and McCrossin in that they, in no way, address the same problems as each other or the problems addressed by Appellant's disclosure. The only relationship Applicant perceives in this cited art is that they were likely coincidentally responsive to the same key word search (both references merely contain the words "filter" and "image").

1. Claims 1, 2, 4-6, 8, and 11-21

In addition, even if the cited combination were appropriate, the Examiner's application of the art is simply inaccurate. In particular, in the Office Action dated 30 March 2007 at 2nd ¶ on page 7, the Examiner rejects claim 1 and cites Olano (Col. 6, lines 50 – 57; and Col. 6, lines 21-23) for the allegedly anticipatory proposition that Olano discloses "a first process requesting a filter from a second process; the related filter and initial image comprising a program." However, Olano discloses nothing of sort, focusing only on the manipulation of program code so that graphics hardware can perform mathematical derivatives. The Olano portion cited by the Examiner discusses estimating the size of an anti-aliasing filter; there is no discussion regarding a request for filters between processes. In reply to Appellant's previous argument on this point (see also, Appellant's remarks in the Reply to Office Action dated 29 Sep. 2006 (filed 28 Dec. 2006) at page 18) the Examiner has made the following unreasonable inference "Since the application performing the first process allows a user to describe images using shading language and the compiler performing the second process transforms shading language statements to run on graphics hardware, the first process requests a filter from a second process." *See* Office Action dated 30 March 2007 at 2nd ¶ on page 7. Thus, other than mentioning a filter, the Olano citation has no reference or relevance to the cited claim language requiring a first process to request a filter from a second process. The Examiner's argument is not Olano's teaching.

The Examiner also cites to Olano Col. 6, lines 21-23, 50-57 to support the statement "One graphics function applied by graphics hardware is the filter that is applied to the image to yield a pixel-image result." At the cited location there is no reference of pixel-image results and as stated above Olano's only filter related reference is to "estimate the size of a filter" which in no way could render Independent claim 1 obvious.

Thus, in view of Appellant's arguments, claim 1 should be allowable. Furthermore, since claims 2, 4-6, 8, and 11-21 all depend from claim 1, Appellant submits that those claims should likewise be allowable.

2. Claims 22, 23, and 25 - 27

In Office Action dated 30 March 2007 ¶ 45 Pg. 25, the Examiner also rejects independent claim 22 over the combination of Olano and McCrossin. Appellant submits that claim 22 is in condition for allowance generally for the reasons stated above and specifically as follows. Furthermore, without waiving argument on all the limitations, Appellant points out the following failures of the Examiner's application of Olano and McCrossin. The Examiner cites Olano's Col. 4, lines 11-21 (see below) as allegedly disclosing "the filter requested by the first process."

> The application program layer comprises application **102**. Application **102** represents any high level application. For example, application **102** might be a program for rendering computer graphics imagery to be used in a movie, or a program for visualization of scientific data. Application **102** will typically allow a user, for example, a movie technical director or a scientist, to describe images or scenes using a high level programming language or a shading language, which causes object files residing at the graphics toolkit layer and/or the graphics API layer to be executed by the graphics system embodying architecture **100**.

As is evident from the Olano portion cited by the Examiner (quoted just above), Olano does not anywhere disclose a first process (or any process) requesting a filter as claimed.

In addition, the Examiner cites the same points in Olano for the claimed first and second microprocessor (Olano Col. 4, lines 11-35 and Col. 4, lines 30-35). While, Col. 4 of Olano may indeed list a variety of exemplary hardware, nothing in Col. 4 or elsewhere teaches the use of a first and second microprocessor according to the claimed division of work. As explained earlier, Olano teaches the use of graphics hardware to perform derivative mathematics. The only attribution of work even implied is that the graphics hardware will do the math. Even the most expansive reading of Olano can not imply, as claimed, a first microprocessor running two interrelated processes where one of those processes causes the creation of a specifically claimed data structure; and then a second microprocessor for running a program created by using the specifically claimed data structure. To the contrary, Olano discusses converting program lines into a node tree and then processing the node tree so that graphics hardware can be used to do

derivative math. Thus, Olano at least lacks both the specifically claimed data structure and the expressly claimed division of work between microprocessors.

Furthermore, the Examiner alleges that Olano's Col. 11, lines 14-18 discloses a memory for storing a pixel image resulting from running a specifically claimed program. And while the cited section does discuss a frame buffer as illustrative hardware, it cannot be, as claimed, a buffer for storing a pixel image resulting from running the claimed program. This is because the only "programs" arguably taught by Olano are for doing math, not for making or editing images. Thus, Olano does not and can not suggest making or editing images using the specifically claimed program.

Moreover, with respect to claim 22, the Examiner asserts that McCrossin teaches a data structure comprising a relationship between an image and a filter. While the Examiner's cited excerpt (Col 6, line 46 – Col. 7, line 9) does discuss "filters" and "images," it most certainly does NOT discuss any data structure, much less the specifically claimed data structure that comprises the claimed relationship between an initial image and a specific filter. See Col 6, line 46 – Col. 7, line 9, which is the Examiner's citation copied directly below.

> API read calls **133** and API write calls **135** are calls to the image readers and writers to control the reading or writing of image data. The API calls employed are defined by the requirements of each particular reader or writer within image readers/writers **137** in accordance with a preferred embodiment of the present invention. Transform object **103** determines an actual image vector **139**, which describes the format of the image to be read or written. Image request vector **127** is received from application **101** and is compared to actual image vector **139** to determine what filters are needed. After such an evaluation, filter stack **141** is constructed to return the image data specified in image request vector **127**.
>
> Filters are accessed by transformation object **103** from filter library **143**, as depicted in FIG. **7**. Filter library **143** contains a number of different filters available for performing various image transformations. In the depicted example, filter library **143** includes the following filters: photometric **143***a*, rotate **143***b*, crop **143***c*, pad **143***d*, scale **143***e*, bit pad **143***f*, dither **143***g*, gray **143***h*, color transform **143***i*, decode **143***j*, and encode **143***k*. These filters are employed by transformation object **103** to provide the necessary transfor-

7

mation or alteration of image data to return image data in a form as specified in image request vector **127**. Transform object **103** selects a number of filters to create a filter stack **141** for use in manipulating the image data. These filters may be used when image readers/writers **137** is employed to read or write data in accordance with a preferred embodiment of the present invention. More information on filters may be found in Foley et al., *Computer Graphics: Principles and Practice* (2d ed. 1991).

Thus, given the lack of any disclosure regarding the claimed data structure comprising the relationship information, McCrossin simply does not apply to the claim language.

In the Examiner's response to Appellant's previous arguments, Office Action dated 30 March 2007 4th ¶ Pg. 9, the Examiner's seems to equate "API" in McCrossin with the claimed "data structure". This is clearly not a valid analogy. After asserting this invalid analogy the Examiner extrapolates McCrossin's "API" asserting that it teaches the claimed "data structure comprising a relationship between said initial image and a specific filter." An API is a generic program interface. It can not teach or suggest the specifically claimed "data structure."

Finally, with respect to claim 22, as explained above, Appellant disagrees with the Examiner's contention that there is any motivation to combine Olano and McCrossin and apply the combination to the claim 22.

Thus, in view of Appellant's arguments, claim 22 should be allowable. Furthermore, since claims 23, and 25 - 27 all depend from claim 22, Appellant submits those claims should likewise be allowable.

3. Claims 28, 29-32 and 36-41

In Office Action dated 30 March 2007 ¶ 50 Pg. 27, the Examiner also rejects independent claim 28 over the combination of Olano and McCrossin. Appellant submits that claim 28 is in condition for allowance generally for the reasons stated above and specifically as follows. Furthermore, without waiving argument on all the limitations, Appellant points out the following failures of Olano and McCrossin with respect to claim 28. The Examiner cites Olano's Col. 2,

lines 45-56 and Col. 4, lines 30-35 (see below) as allegedly disclosing "a second process servicing the requests of the first process, the servicing comprising the steps of optimizing a graph representing the result image; compiling the optimized graph; causing the rendering of the complied optimized graph."

tems. In an embodiment, a data structure representing the operation of one or more computer program statements is received by a compiler and transformed into a tree data structure. Nodes of the tree data structure that determine derivative information are marked. A transform rule is applied to the marked nodes to transform the tree data structure into a data structure that can be implemented with graphics hardware interface program statements. In one embodiment of the invention, the compiler transforms shading language statements into graphics application programming interface statements that can be implemented with multiple passes through a graphics pipeline. It is a feature of

Alternatively, compiler **104** can be implemented so that it will transform high level statements into statements designed to take advantage of specialized graphics hardware available in a particular graphics system, thereby optimally transforming high level statements to run on a particular graphics system.

As is evident from the Examiner's citations quoted above, Olano merely proposes making a node tree from program statements and then adapting the tree to perform mathematical derivatives on graphics hardware. Olano doesn't even purport to do so in the context of a second process servicing the request of a first process and Olano doesn't mention or suggest optimizing anything (Olano's adaptation of the node tree for math and the graphics processor is not and does not purport to be anything like optimization).

The Examiner also cites McCrossin (Col. 5, lines 49-50, Col. 6, lines 51-58, and lines 9-21) to allegedly anticipate a first process requesting creation of a context, the first process indicating parameters associated with the creation of the result image; and the first process requesting that the result image be rendered to the context. However, the cited McCrossin

excerpts merely discuss a transform object selection of vectors for McCrossin's format conversions. This has nothing to do with creating contexts and rendering to same. McCrossin is focused on the conversion of formats and thus assumes the cooperation of some lower level resource accommodations; however, McCrossin does not speak to those accommodations or any techniques for effecting same. In sum, McCrossin is totally unconcerned with the notion of context creation or use, much less in the specifically claimed manner.

In Office Action dated 30 March 2007 2nd ¶ Pg. 10, the Examiner cites Olano (Col. 5, lines 35-36, 56-60) to allegedly anticipate "optimizing a graph representing the result image." However, the Examiner has taken Olano's words "optimally transforming" out of context to equate the claimed "optimization." Olano Col. 5 lines 46-51 put the Examiner's citation in the correct context. That reference clearly describes that Olano's "optimally transforming" as a feature whereby Olano's compiler "can be quickly and flexibly reconfigured to operate with a different set of high level programming statements or shading language statements or with different graphics hardware." Thus, Olano's "optimal transformation" is a result of editing a text file of configuration rules in order to adapt Olano's compiler. This is clearly not the optimization Appellant claims.

Finally, with respect to claim 28, as discussed specifically above, Appellant disagrees with the Examiner's contention that there is any motivation to combine Olano and McCrossin.

Thus, in view of Appellant's arguments, Appellant submits that claim 28 should be in condition for allowance. Furthermore, since claims 29-32 and 36-41 all depend from claim 28, Appellant submits those claims should likewise be allowable.

4. Claims 43-47 and 51-56

In Office Action dated 30 March 2007 ¶ 58 Pg. 29, the Examiner rejects claim 43 with citation back to the Examiner's arguments for claim 28; except the Examiner suggests that claim 43 has the additional limitation of a first process and a second process running on a first microprocessor and rendering occurring on a second microprocessor. The Examiner alleges

anticipation of this limitation citing Olano at Col.3, lines 48-53 and Col. 4, lines 30-35; both excerpts copied directly below.

> tems. In one embodiment of the invention, a compiler transforms shading language statements into graphics API statements that can be implemented with multiple passes through a graphics pipeline. It is a feature of the present invention that it can be implemented on a wide range of available computer graphics systems including, but not limited to, a general purpose computer having a graphics processor (single chip or multiple chip), a specially designed graphics computer, a graphics machine, a parallel processing graphics machine, a graphics card, graphics hardware, a graphics accelerator, a computer network architecture, et cetera.

> maximum compatibility with graphics systems. Alternatively, compiler **104** can be implemented so that it will transform high level statements into statements designed to take advantage of specialized graphics hardware available in a particular graphics system, thereby optimally transforming high level statements to run on a particular graphics system.

No reasonable reading of these excerpts can find any discussion or teaching regarding the claimed division of work between two microprocessors. These passages do not even suggest any specific division of work between processors of any kind, much less the specifically claimed division of work.

Furthermore, with respect to claim 43, Appellant refers to arguments made previously with respect to claim 28. Thus, in view of Appellant's arguments, Appellant submits that claim 43 is in condition for allowance. Furthermore, since claims 44-47 and 51-56 all depend from claim 43, Appellant submits those claims should likewise be allowable.

5. Claims 58, 61-62

In Office Action dated 30 March 2007 ¶ 60 Pg. 30, the Examiner also rejects independent claim 58 over the combination of Olano and McCrossin. Appellant submits that claim 58 is in

condition for allowance generally for the reasons stated above and specifically as follows. Furthermore, without waiving argument on all the limitations, Appellant points out the following failures of the application of Olano combined with McCrossin. The Examiner cites Olano's Col. 10, lines 59-65 (see below) as allegedly disclosing an API call for the function of creating an image.

> Rasterization module **814** combines the output of vertex operation module **812** and pixel operation module **820** to form image fragments. Image fragments correspond to pixels in frame buffer **818**. Image fragments have both color values, such as red, green, blue, and alpha, and a depth value, as would be known to a person skilled in the relevant art.

As is evident from the cited excerpt reproduced above, there is no teaching or suggestion of an API call having the function to create an image. Furthermore, while Olano may propose the concept of an image it does NOT propose, teach or suggest the claimed step of ***associating*** an image or anything else with a with the claimed function call. Once again, Olano is a system to perform math with graphics hardware, thus Appellant's specifically crafted claims regarding a high-level graphics API are far removed from anything in the Olano disclosure. Furthermore, the Examiner's citation to McCrossin is similarly misplaced because, as explained earlier, McCrossin's only APIs are to translate the same image from one format to another and thus McCrossin offers nothing in terms of a graphics API, much less defining relationships between such APIs and the specifically claimed objects.

Finally, with respect to claim 58, as discussed specifically above, Appellant disagrees with the Examiner's contention that there is any motivation to combine Olano and McCrossin

Thus, in view of Appellant's arguments, Appellant submits that claim 58 should be in condition for allowance. Furthermore, since claims 61-62 depend from claim 58, Appellant submits those claims should likewise be allowable.

6. Claims 63, 64, 66, and 67

In Office Action dated 30 March 2007 ¶ 63 on Pg. 31, the Examiner has rejected claims 63, 64, 66, and 67 making only reference to the Examiner's arguments with respect to claims 58, 59, 61 and 62. Referring back and forward in this paper to Appellant's respective arguments, Appellant submits these claims are in condition for allowance.

Finally, with respect to independent claims 63, 64, 66 and 67, as discussed specifically above, Appellant disagrees with the Examiner's contention that there is any motivation to combine Olano and McCrossin.

Thus, in view of Appellant's arguments, Appellant submits that claim 63 should be in condition for allowance. Furthermore, since claims 64, 66-73 all depend from claim 63, Appellant submits those claims should likewise be allowable.

7. Claim 74

In Office Action dated 30 March 2007 ¶ 70 Pg. 32, the Examiner also rejects independent claim 74 over the combination of Olano and McCrossin. Appellant submits that claim 74 is in condition for allowance generally for the reasons stated above and specifically as follows. Furthermore, without waiving argument on all the limitations, Appellant points out the following failures in the application of Olano combined with McCrossin. As discussed above, Olano certainly does not disclose optimizing a graph representing an image and the Examiner's citation to Col. 4, lines 30-36 does not even suggest same:

> Alternatively, compiler **104** can be implemented so that it will transform high level statements into statements designed to take advantage of specialized graphics hardware available in a particular graphics system, thereby optimally transforming high level statements to run on a particular graphics system.

Furthermore, Olano does not disclose any division of work between processing entities, much less the specifically claimed division of work between a CPU and GPU. The Examiner

has cited the same excerpt directly above for this anticipation allegation; and no reasonable reading to the passage or anything else in Olano can infer the claimed division of work. Furthermore, the Examiner cites McCrossin for alleged anticipation regarding rendering to a specified context. However, the cited excerpt merely discusses McCrossin's transform object's selection of vectors for McCrossin's format conversions. This has nothing to do with creating contexts and rendering to contexts. McCrossin is focused on the conversion of formats and thus assumes the cooperation of some lower level resource accommodations; however, McCrossin does not speak to those accommodations or any techniques for effecting same. In sum, McCrossin is totally unconcerned with the notion of context creation or use, much less in the specifically claimed manner

Finally, with respect to claim 74, as discussed specifically above, Appellant disagrees with the Examiner's contention that there is any motivation to combine Olano and McCrossin.

Thus, in view of Appellant's arguments, Appellant submits that claim 74 should be in condition for allowance.

8. Claim 85

In Office Action dated 30 March 2007 ¶ 71 Pg. 33, the Examiner rejects claim 85. Appellant submits that claim 85 is in condition for allowance because claim 85 depends upon other claims which have been demonstrated to overcome the cited art.

D. Rejections Based Upon The Three-Way Combination Of Olano, Mccrossin And Levy

1. Claims 3, 33-35, and 48-50

In Office Action dated 30 March 2007 ¶ 72 Pg. 33, the Examiner rejects claims 3, 33-35, and 48-50 as being unpatentable over the 3-way combination of Olano, McCrossin and Levy (US 20020033844A1). The Examiner relies on Olano to teach "optimization" and for Levy to include "the step of using a cache look-up". As clearly described above the Examiners citation from Olano for optimization was taken out of context and does not teach the claimed optimization. Therefore, the combination of Olano and Levy can not teach or suggest every

aspect of Appellant's claims. Additionally, claim 3 depends from claim 1 which has been demonstrated to overcome the cited art; claims 33-35 depend from claim 28, which has been demonstrated to overcome the cited art; and claims 48-50 depend from claim 43, which has been demonstrated to overcome the cited art.

Finally, with respect to claims 3, 33-35, and 48-50, as discussed specifically above, Appellant disagrees with the Examiner's contention that there is any motivation to combine these three widely ranging references.

Thus, Appellant submits that claims 3, 33-35 and 48-50 should be in condition for allowance.

E. Rejections Based Upon The Three-Way Combination Of Olano, Mccrossin And Parikh

1. Claims 7, 9, 60 and 65

In Office Action dated 30 March 2007 ¶ 78 Pg. 35, the Examiner rejects claims 7, 9, 60 and 65 as being unpatentable over the 3-way combination of Olano, McCrossin and Parikh (US 6411301). The Examiner relies on Olano and McCrossin for the teachings relative to claim 1. In the discussion above Appellant has clearly demonstrated that Olano and McCrossin do not teach or suggest Appellant's claim 1. Additionally, The Examiners relies on Parikh to teach "a compiled program for yielding a pixel-image result" with the citation to Parikh as follows:

> "system that pre-compiles the lists in preparation for use by the graphics and audio processor 114. Such display lists can be stored by any storage device within system 50 or accessible to it, including but not limited to memory card 144, flash memory 140, boot ROM 134, audio memory 126, etc." Parikh (Col. 10, lines 1-5)

The Examiner has again taken a citation out of context because Parikh is pre-compiling a **list** of "display lists and/or audio lists" Parikh (Col. 9, lines 66-67). This is not "compiling" in the computer science sense. The pre-compiling in Parikh does not yield a program and certainly has nothing to do with a pixel-image result, as in Appellant's claim. Appellant additionally notes

that because: claims 7 and 9 depends from claim 1 which has been demonstrated to overcome the cited art; claims 60 depends from claim 58, which has been demonstrated to overcome the cited art; and claim 65 depends from claim 63, which has been demonstrated to overcome the cited art.

Finally, with respect to claims 7, 9, 60 and 65, as discussed specifically above, Appellant disagrees with the Examiner's contention that there is any motivation to combine these three widely ranging references.

Thus, Appellant submits that claims 7, 9, 60 and 65 should be in condition for allowance.

F. Rejections Based Upon The Four-Way Combination Of Olano, Mccrossin, Parikh And Doyle

1. Claim 10

In Office Action dated 30 March 2007 ¶ 83 Pg. 36, the Examiner rejects claim 10 as being unpatentable over the 4-way combination of Olano, McCrossin, Parikh and Doyle (US 6867779). Appellant requests reversal of this rejection because claim 10 depends from claim 7, and claim 7 depends from claim 1. Both claims 7 and 1 of have been discussed in detail above and are clearly demonstrated to overcome the cited art.

Finally, with respect to claim 10, as discussed specifically above, Appellant disagrees with the Examiner's contention that there is any motivation to combine these four widely ranging references. Indeed, while Appellant disagrees with the Examiner's application of the art to the claims, even if correctly applied, the combination of these 4 items of art would be independently inventive, i.e. sufficiently novel and non-obvious under 35 United States Code.

Thus, Appellant submits that claim 10 should be in condition for allowance.

G. Rejections Based Upon The Three-Way Combination Of Olano, Mccrossin And Sturges

1. Claim 24

In Office Action dated 30 March 2007 ¶ 84 Pg. 37, the Examiner rejects claim 24 as being unpatentable over the three-way combination of Olano, McCrossin and Sturges (US

5854637). The Examiner relies on Sturges to teach "a graphics API" with the following reference:

> "drawing engine software package 238. The drawing engine software of the CPU may utilize conventional rendering packages such as **3DR by API software** or Open Graphics Language (OpenGL), a standard graphics software library." (Sturges at Col. 11, lines 42-45) emphasis added.

Appellant submits that the above reference to an API is in fact a company name and can in no way teach an API (Application Program Interface) or be, as the Examiner contends, a reference related to Olano merely because both references happen to use the same 3 letters. Appellant additionally notes that claim 24 depends from claim 22, which has been demonstrated to overcome the cited art.

Finally, with respect to claim 24, as discussed specifically above, Appellant disagrees with the Examiner's contention that there is any motivation to combine these three widely ranging references. Thus, Appellant submits that claim 24 should be in condition for allowance.

H. Rejections Based Upon The Three-Way Combination Of Olano, Mccrossin And Stokes

1. Claims 42 and 57

In Office Action dated 30 March 2007 ¶ 85 Pg. 38, the Examiner rejects claims 42 and 57 as being unpatentable over the three-way combination of Olano, McCrossin and Stokes (US 6977661). Appellant requests reversal of this rejection because: claim 42 depends from claim 28 which has been demonstrated to overcome the cited art; and claims 57 depends from claim 43, which has been demonstrated to overcome the cited art.

Finally, with respect to claims 42 and 57, as discussed specifically above, Appellant disagrees with the Examiner's contention that there is any motivation to combine these three widely ranging references.

Thus, Appellant submits that claims 42 and 57 should be in condition for allowance.

I. Rejections Based Upon The Three-Way Combination Of Olano, Mccrossin And Doyle

1. Claim 59

In Office Action dated 30 March 2007 ¶ 88 Pg. 39, the Examiner rejects claim 59 as being unpatentable over the three-way combination of Olano, McCrossin and Doyle. Appellant requests reversal of this rejection because claim 59 depends from claim 58, which has been demonstrated to overcome the cited art.

Finally, with respect to claim 59, as specifically discussed above, Appellant disagrees with the Examiner's contention that there is any motivation to combine these three widely ranging references.

Thus, Appellant submits that claim 59 should be in condition for allowance and reversal of the rejection is respectfully requested.

J. Rejections Based Upon The Two-Way Combination Of Olano And Levy

1. Claims 76 and 77

In Office Action dated 30 March 2007 ¶ 89 Pg. 40, the Examiner rejects claims 76 and 77 as being unpatentable over the combination of Olano and Levy (US 20020033844A1). Appellant requests reversal of this rejection because claims 76 and 77 depend from claim 75, which has been demonstrated to overcome the cited art.

Finally, with respect to claims 76 and 77, Appellant disagrees with the Examiner's contention that there is any motivation to combine these two non-analogous references. As discussed earlier, Levy proposes a method for connecting multimedia content to a network resource, while Olano proposes a system for performing derivative math with a graphics processor. There is simply no reason combine references that differ as much as these.

Thus, Appellant submits that claims 76 and 77 should be in condition for allowance.

K. Rejections Based Upon The Two-Way Combination Of Olano And French

1. Claim 78

In Office Action dated 30 March 2007 ¶ 92 Pg. 41, the Examiner rejects claim 78 as being unpatentable over the combination of Olano and French (US 6266053). Appellant submits that claim 78 is in condition for allowance because claim 78 depends from claim 75 which has been demonstrated to overcome the cited art.

Finally, with respect to claim 78, as discussed specifically above, Appellant disagrees with the Examiner's contention that there is any motivation to combine these two non-analogous references.

L. Conclusion

Appellant has addressed all of the Examiner's rejections and has shown a reason for reversal of each of these rejections. The Examiner has combined eight different references to contend that they disclose Appellant's claims. This is simply not the case. Each of these references, even when taken together, do not disclose each and every limitation of Appellant's claims. In fact the references are taken out of context and then unreasonably combined with other non-applicable references. The cited prior art simply can not support a rejection based on anticipation or obviousness. In fact, if there were a reasonable combination of these eight references, the combination would indeed be inventive.

In summary, Appellant traverses all of the Examiner's art-based rejections because the primary references used by the Examiner are plainly inapplicable to the inventions claimed by Appellant. Furthermore, in addition to alleging the application of distant and literally inapplicable art, the Examiner's specific ***allegations*** regarding each claim limitation do not incorporate the claimed connectivity between claim elements. For example, with respect to Claim 22, the Examiner has alleged McCrossin teaches that a data structure comprises a relationship between an image and a filter. While the Examiner's cited excerpt (Col 6, line 46 – Col. 7, line 9) does discuss "***filters***" and "***images***," it most certainly does NOT discuss any ***data structure comprising a relationship between an initial image and a specific filter.*** The

Examiner's rejections must fail for lack of even allegations regarding Appellant's claimed connectivity between elements.

With regard to all the Section 103 rejections, in addition to misapplying the art, the Examiner has failed to show any motivation to combine references. For example, the Examiner's two primary references are Olano and McCrossin. The Examiner's primary reference is Olano, which relates to system for using graphics hardware to perform derivative math. This reference is obscure to the mainstream of graphics processing and image processing, thus non-analogous in every sense. The Examiner also relies heavily on the McCrossin reference, which relates to using the sequential application filters to change the file format of an image (e.g. from BMP to JPG, etc.). A skilled artisan working in the field of the Appellant's claims would not likely be inclined to find even one of these references. But, even if one were found (accidentally), since the topic of the other is not analogous to anything else, there would be no motivation to find or combine the two. This is particularly so for the purpose of developing the claimed invention. In essence, the combination of Olano and McCrossin "teaches" the non-sensical technical point of graphics systems that convert file formats in order to do derivative calculus.

Appellant submits that Olano alone or in any combination with McCrossin, Levy, Doyle, Sturges, Stokes, French or Parikh fail to teach each recited element in the rejected claims. Accordingly, Appellant respectfully requests the Board reverse the Examiner's rejections.

Respectfully submitted,

Lou Brucculeri, J.D.

Reg. No. 38,834

Wong, Cabello, Lutsch, Rutherford & Brucculeri, L.L.P.
Customer No. 61947
Voice: 832-446-2415
20333 SH 249,
Suite 600
Houston, Texas 77070
Facsimile: 832-446-2458
Email: lbrucculeri@counselIP.com

VIII. CLAIMS APPENDIX

We claim:

1. (Previously Presented) A method of editing an initial image, comprising the steps of:
   - A first process requesting a filter from a second process;
   - Said first process defining a relationship between said filter and said initial image, said related filter and initial image comprising a program,
   - Said second process compiling said program, yielding a compiled program;
   - Running at least a portion of said compiled program to apply a function of said filter to said image, yielding an pixel-image result.
2. (Original) The method of claim 1 having the additional step of
   - Optimizing said program.
3. (Original) The method of claim 2, wherein the step of optimizing includes the step of using a cache look-up to see if said pixel-image result is already in cache.
4. (Original) The method of claim 2, wherein the step of optimizing includes the step of calculating an intersection, said intersection representing an area where said pixel-image result is both defined and in a result region requested of said second process.
5. (Previously Presented) The method of claim 4 further comprising the step of using said calculated intersection to limit the number of pixels that require calculation during said running of said compiled program.
6. (Original) The method of claim 4 further comprising the step of, using said calculated intersection to limit the amount of memory necessary for storing calculated images.
7. (Original) The method of claim 1 wherein said compiled program is for a single microprocessor.
8. (Original) The method of claim 1 wherein said compiled program comprises a component for a first microprocessor and a component for a second microprocessor.
9. (Original) The method of claim 7 wherein said single microprocessor is a CPU.

10. (Original) The method of claim 7 wherein said single microprocessor is a programmable GPU.
11. (Original) The method of claim 8 wherein said first microprocessor is a CPU and said second microprocessor is a GPU.
12. (Original) The method of claim 8 wherein said first and second microprocessors are both CPUs.
13. (Original) The method of claim 8 wherein said first and second microprocessors are both GPUs.
14. (Original) The method of claim 1 wherein said initial image is only a color.
15. (Original) The method of claim 1 wherein said first process is an application program.
16. (Original) The method of claim 1 wherein said second process comprises a suite of graphics services functions.
17. (Original) The method of claim 1 wherein an operating system comprises said second process.
18. (Original) The method of claim 1 wherein said first process requests a high-level filter and said second process responds with an object representing a lower-level filter.
19. (Original) The method of claim 1 wherein said first process and second process run on a CPU and said compiled program runs on a GPU.
20. (Original) The method of claim 2 wherein said first process and second process run on a CPU and said compiled program runs on a GPU.
21. (Original) The method of claim 5 wherein said first process and second process run on a CPU and said compiled program runs on a GPU.
22. (Original) A system for editing an initial image, comprising:
    - A first microprocessor running a first process and a second process for servicing requests from said first process;
    - A memory for storing a filter, said filter requested by said first process;
    - A second memory for storing a data structure comprising a relationship between said initial image and said filter, said first process causing the creation of said data structure;
    - A second microprocessor for running a program created using said data structure;

- A third memory for storing a pixel image resulting from running said program.

23. (Original) The system of claim 22 wherein said first and second memories are the same.

24. (Previously Presented) The system of claim 22 wherein said first, second and third memories are the same.

25. (Original) The system of claim 22 wherein said first and second memories are in system memory and said third memory is in video memory.

26. (Original) The system of claim 22 wherein said first microprocessor processes said data structure to produce said program for use on said second microprocessor.

27. (Original) The system of claim 22 wherein said second microprocessor, under control of said program, causes said pixel image to be stored in said third memory.

28. (Previously Presented) A method of creating a result image comprising the steps of:

- a first process requesting the creation of a context;
- said first process requesting the creation of a result image;
- said first process indicating parameters associated with the creation of said result image;
- said first process requesting that said result image be rendered to said context;
- a second process servicing said requests of said first process, said servicing comprising the steps of:
  - optimizing a graph representing said result image;
  - compiling said optimized graph;
  - causing the rendering of said compiled optimized graph into said context.

29. (Original) The method of claim 28 wherein said creation of said result image comprises editing a pre-existing image.

30. (Previously Presented) The method of claim 28 wherein said step of optimizing a graph representing said first image comprises the step of:

- Calculating an intersection, said intersection representing an area where said result image is both defined and requested as a result by said first process.

31. (Original) The method of claim 28 wherein said step of optimizing a graph representing said first image comprises the step of:

- Analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.

32. (Original) The method of claim 30 wherein said step of optimizing a graph representing said first image comprises the additional step of:
    - Analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.
33. (Original) The method of claim 31 wherein the step of analyzing adjacent nodes comprises the step of checking a cache to determine if the result of such analysis is available in a memory.
34. (Original) The method of claim 28 wherein the step of optimizing said graph comprises the step of checking a cache to determine if said graph has already been optimized.
35. (Original) The method of claim 28 wherein the step of optimizing said graph comprises the step of checking a cache to determine if the result of rendering said graph is available in a memory.
36. (Original) The method of claim 28 wherein said first process requests the output of a graph programmatically assembled by said first process and comprising one or more pre-defined filters.
37. (Original) The method of claim 36 wherein said first process programmatically assembled said graph in cooperation with said second process.
38. (Original) The method of claim 28 wherein said first process is an application program.
39. (Original) The method of claim 28 wherein said second process comprises a suite of graphics services functions.
40. (Original) The method of claim 39 wherein an operating system comprises said second process.
41. (Original) The method of claim 28 wherein an operating system comprises said second process.
42. (Original) The method of claim 28 wherein said second process inserts nodes into said graph for converting an original color scheme to an operating color scheme and for converting the operating color scheme back to the original color scheme.

43. (Original) A method of creating an result image comprising the steps of:
    - a first process running on a first microprocessor requesting the creation of a context;
    - said first process requesting the creation of a result image;
    - said first process indicating parameters associated with the creation of said result image;
    - said first process requesting that said result image be rendered to said context;
    - a second process running on said first microprocessor servicing said requests of said first process, said servicing comprising the steps of:
        - optimizing a graph representing said result image;
        - compiling said optimized graph;
        - causing rendering of said compiled optimized graph into said context, said rendering occurring on a second microprocessor.
44. (Original) The method of claim 43 wherein said creation of said result image comprises editing a pre-existing image.
45. (Original) The method of claim 43 wherein said step of optimizing a graph representing said first image comprises the step of:
    - Calculating an intersection, said intersection representing an area where said result image result is both defined and requested by said first process.
46. (Original) The method of claim 43 wherein said step of optimizing a graph representing said first image comprises the step of:
    - Analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.
47. (Original) The method of claim 45 wherein said step of optimizing a graph representing said first image comprises the additional step of:
    - Analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.
48. (Original) The method of claim 46 wherein the step of analyzing adjacent nodes comprises the step of
    - checking a cache to determine if the result of such analysis is available in a memory.

49. (Original) The method of claim 43 wherein the step of optimizing said graph comprises the step of
    - checking a cache to determine if said graph has already been optimized.
50. (Original) The method of claim 43 wherein the step of optimizing said graph comprises the step of
    - checking a cache to determine if the result of rendering said graph is available in a memory.
51. (Original) The method of claim 43 wherein said first process requests the output of a graph programmatically assembled by said first process and comprising one or more pre-defined filters.
52. (Original) The method of claim 51 wherein said first process programmatically assembled said graph in cooperation with said second process.
53. (Original) The method of claim 43 wherein said first process is an application program.
54. (Original) The method of claim 43 wherein said second process comprises a suite of graphics services functions.
55. (Original) The method of claim 54 wherein an operating system comprises said second process.
56. (Original) The method of claim 43 wherein an operating system comprises said second process.
57. (Original) The method of claim 43 wherein said second process inserts nodes into said graph for converting an original color scheme to an operating color scheme and for converting the operating color scheme back to the original color scheme.
58. (Previously Presented) A method for providing a high level interface to a graphics processing resource comprising:
    - A first process requesting performance of a task through one or more function calls serviced by said graphics processing resource, said function calls selected from one or more of the following options, (i) creating of an image, (ii) creating a filter, (iii) setting arguments associated with said filter, (iv) asking for the output of said filter or another filter, (v) creating a context; and (vi) rendering an image to said context or another context;

- Said request having an object associated therewith, said object comprising one of the following: a context, an image, a filter, or a vector;
- Said request defining a relationship between at least one of said functions and one of said objects,
- Said graphics processing resource servicing said request.

59. (Original) The method of claim 58 wherein said request is serviced using a GPU and a CPU.

60. (Original) The method of claim 58 wherein said request is serviced using an emulator to run a GPU program on a CPU.

61. (Original) The method of claim 58 comprising the additional step of creating a graph representing an image.

62. (Original) The method of claim 61 comprising the additional step of optimizing said graph.

63. (Previously Presented) A method for providing a high level interface to a graphics processing resource comprising:

- A first process requesting performance of a task through one or more function calls serviced by said graphics processing resource, said function calls selected from one or more of the following options, (i) creating of an image, (ii) creating a context; and (v) rendering an image to said context or another context;
- Said request having an object associated therewith, said object comprising one of the following: a context, or an image
- Said request defining a relationship between at least one of said functions and one of said objects,
- Said graphics processing resource servicing said request.

64. (Original) The method of claim 63 wherein said request is serviced using a GPU and a CPU.

65. (Original) The method of claim 63 wherein said request is serviced using an emulator to run a GPU program on a CPU.

66. (Original) The method of claim 63 comprising the additional step of creating a graph representing an image.

67. (Original) The method of claim 66 further comprising the step of optimizing said graph.

68. (Original) The method of claim 63, wherein said function calls include the option of creating a filter.

69. (Original) The method of claim 63 wherein said function calls include the option of setting arguments associated with said filter.

70. (Original) The method of claim 68 wherein said function calls include the option of setting arguments associated with said filter

71. (Previously Presented) The method of claim 63 wherein said another object associated with said request may comprise a filter.

72. (Original) The method of claim 63 wherein another object associated with said request may be a vector.

73. (Original) The method of claim 71 wherein another object associated with said request may be a vector.

74. (Previously Presented) A method for rendering an image comprising the steps of:
- A first process running on a CPU requesting the creation of an image;
- A graphics services resource, responding to said request by running a first routine on said CPU, said first routine for optimizing a graph representing said image;
- Said first process requesting the rendering of said image to a specified context;
- Said graphics services resource causing the rendering of said graph representing said image, said rendering occurring on a GPU.

75. (Previously Presented) A method for converting a first image representation into a second image representation, comprising the steps of:
- Creating a first graph associated with said first image representation where software routines for creating such graph execute on a CPU,
- Determining an intersection of the first graph's global domain of definition and global region of interest;
- Resolving a first node in said first graph by running software routines on said CPU to (i) determine if said first node may be combined with a second node and (ii) create program steps for calculating and storing only the portions of any intermediary image

that relate to said intersection, said program steps for compilation on said CPU and execution on a GPU.

76. (Original) The method of claim 75 wherein the step of determining if said first node may be combined with said second node comprises the step of checking a cache to determine if there is a result in memory regarding such determination regarding combining nodes.

77. (Original) The method of claim 75 wherein the step of resolving said first node comprises the step of checking a cache to determine if there is a result in memory regarding the resolution of said first node.

78. (Original) The method of claim 75 wherein said first node is a root node.

79. (Original) An image processing application program interface embodied on one or more computer readable media, comprising: a first group of services related to filter objects; a second group of services related to image objects; a third group of services related to context objects; and a fourth group of services related to vector objects.

80. (Original) The image processing application program interface of claim 79, wherein the first group of services comprise: first functions to create filter objects; second functions to set filter object parameters; and third functions to obtain filter object output.

81. (Original) The image processing application program interface of claim 79 wherein the second group of services comprise: first functions to create image objects; and second functions to render an image object to a context object.

82. (Original) The image processing application program interface of claim 79, wherein the third group of services comprise first functions to create context objects.

83. (Original) The image processing application program interface of claim 79 wherein the fourth group of services comprise: first functions to create vector objects.

84. (Original) An application program interface for facilitating image processing, the application program interface comprising functions to: create image objects; create context objects; create filter objects; set filter object parameters; obtain filter object output; and convert image objects into context objects.

85. (Original) A computer-readable medium having computer executable instructions for performing the method recited in any one of claims 1, 28, 43, 58, 63, 74 or 75.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.